Next Gen IO

Scalable
Scientific Data Management

Carlos Maltzahn,
Systems Research Lab, UC Santa Cruz,
Institute for Scalable Scientific Data Management, LANL/UCSC
Ultrascale Systems Reserch Center, NM Consortium

Next Gen: what is ***not*** going to change

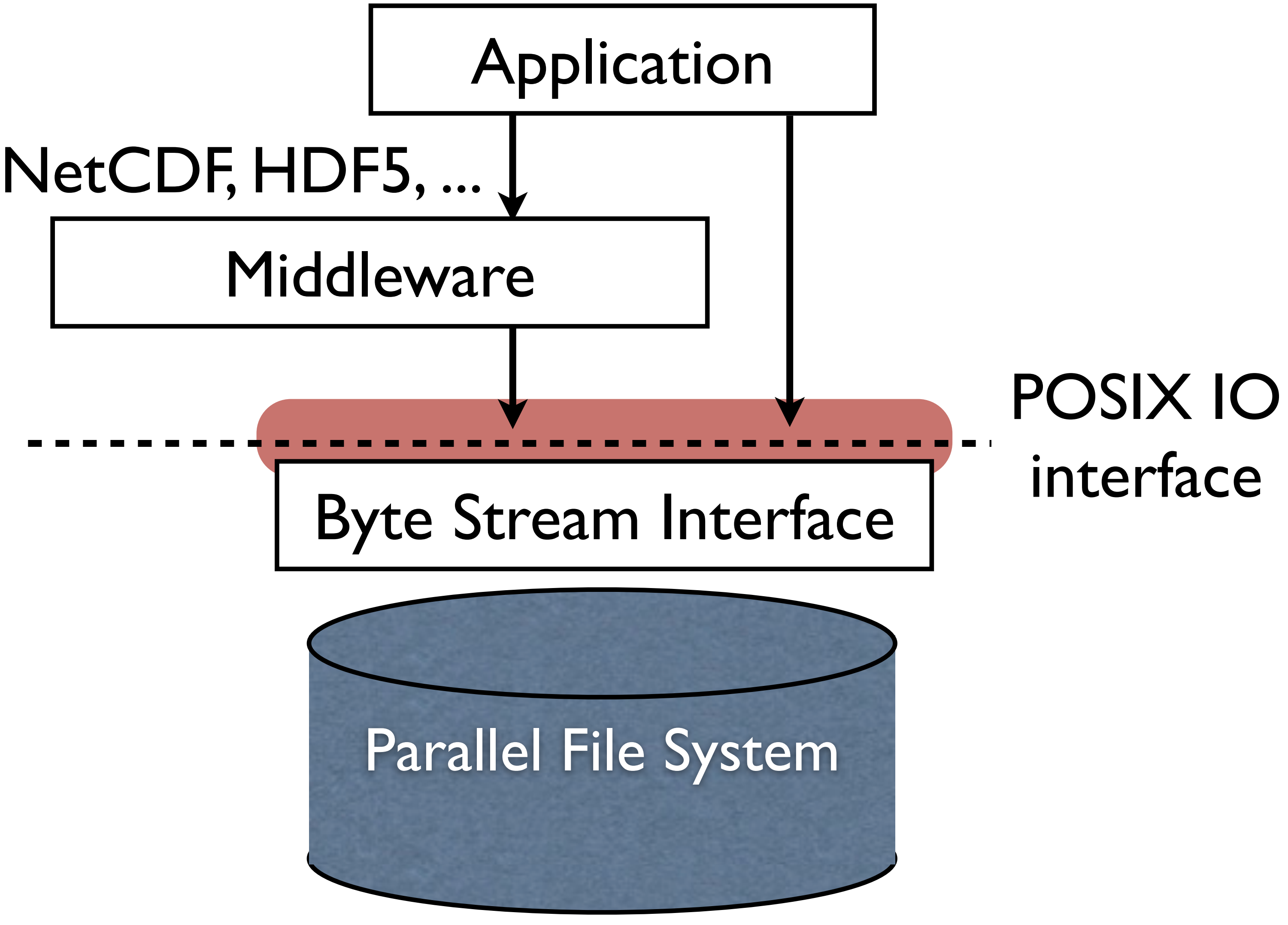
- Digital Data will continue to
  - grow **exponentially**
  - require **active** protection
  - outgrow **read speed** of archival storage media
  - consume a lot of **power**
  - be stored in **byte streams**
  - be hard to move or **convert**

Next Gen: what *is* going to change

- Data access will rely on
  - data **structure**
    - Parsing overhead of unstructured data unaffordable
    - Examples: Apache Avro, Binary XML, ProtocolBuffers, Multimedia, ...
  - **temporal** structure
    - Applications do have utilization needs and deadlines: specify them!
  - well-known data **models**
    - Allows automatic access optimization
    - Minimizes data movement (due to shared model)
  - **automatic** access optimization
    - Allows declarative querying, updates
    - User won't want to re-invent optimization for each application

The POSIX I/O Bottleneck

- **POSIX IO** dominates File system interface
- POSIX IO does not scale
  - **50 years ago**: 100MB
  - **Now**: 100PB (x 1 billion)
- Performance price of POSIX IO is high
  - Workload- & system-specific interposition layers (e.g. PLFS): almost **1,000 x speed-up**
- Common Workaround
  - **Middleware** tries to make up for limitations
  - Still uses POSIX!

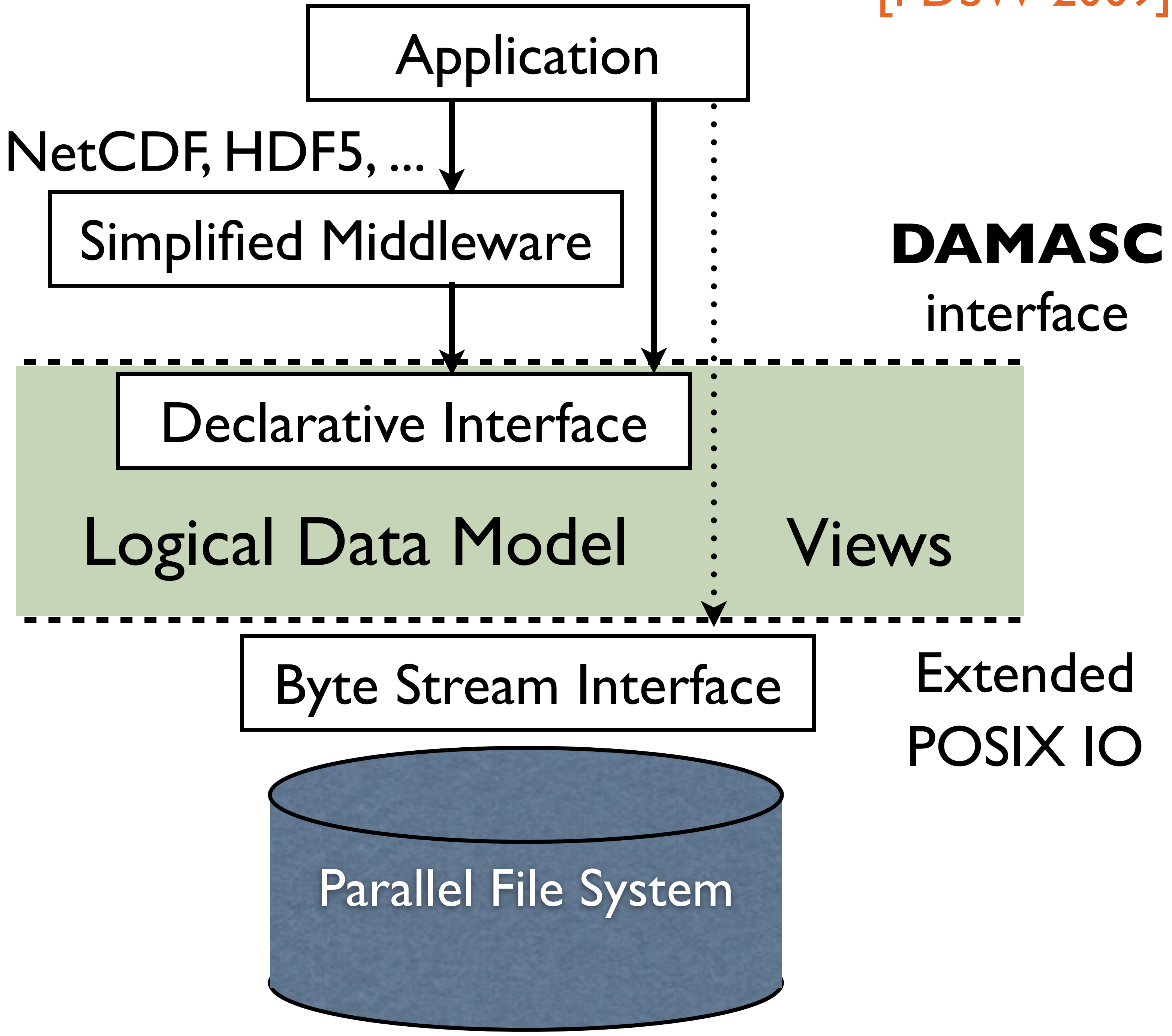


Heavy! Do not move!

DAMASC: DAta MAnagement in Scientific Computing

[PDSW 2009]

- Enhance parallel file system with data services
  - Declarative **querying**
  - **Views**
  - Automatic content **indexing**
  - **Provenance** tracking
- **Index**, not ingest!
- *In situ* **processing** on storage nodes



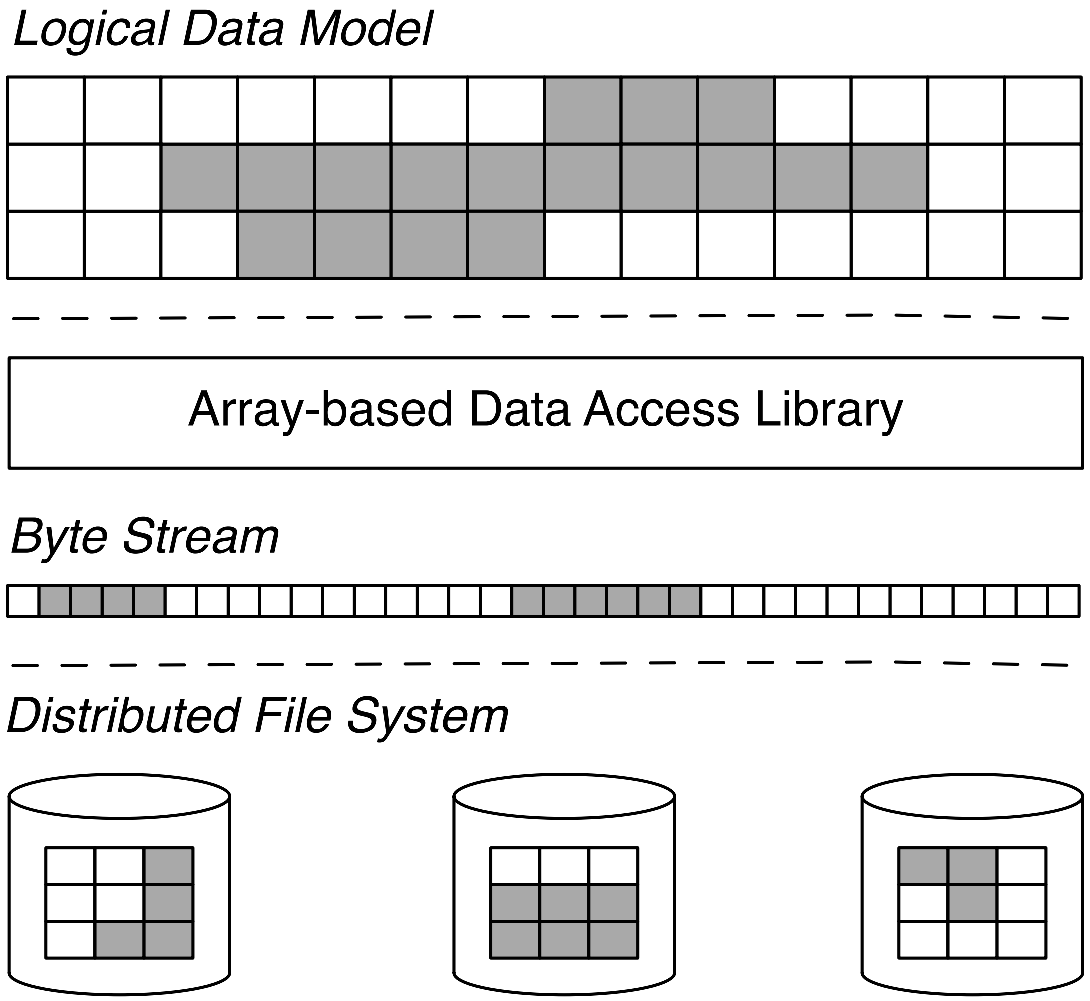
Heavy! Do not move!

DAMASC: SciHadoop [SC11]

- All access via **scientific access library** (e.g. NetCDF)
- Task manager **partitions** logical space
  - instantiates mappers and reducers for logical partition
  - **places** mappers and reducers based on logical relationships
- Benefits of structure-awareness
  - reduces **data transfers**
  - reduces **remote reads**
  - reduces **unnecessary reads**

Scientific File Formats

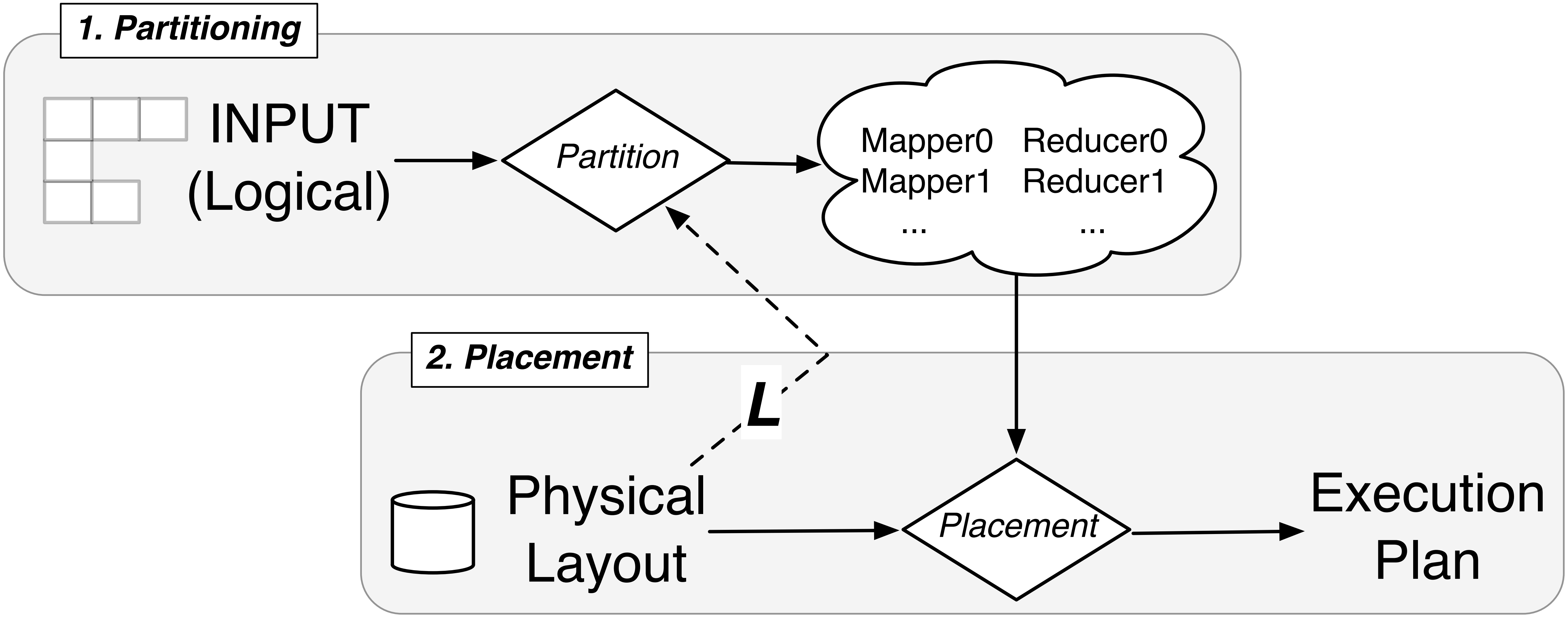
- High-level logical data model (e.g. arrays)
- Translates logical view to physical locations
- All data access must pass through the access library
- Library hides data location



Query Language

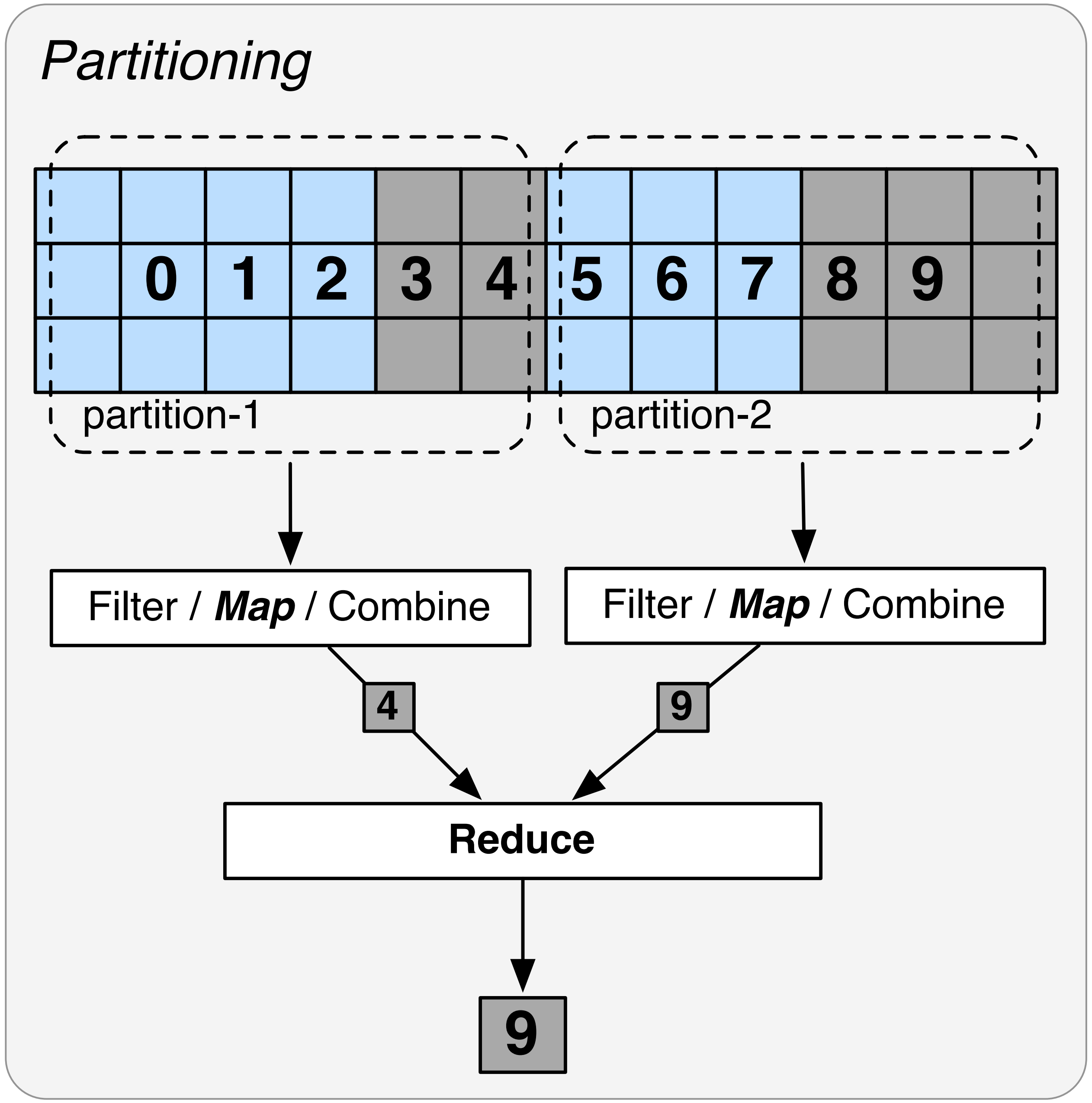
- Details in SC11 paper
- Functions applied to arrays
    - *What is the maximum value in some array?*
- Simple language exposes data requirements

SciHadoop Partitioning and Placement

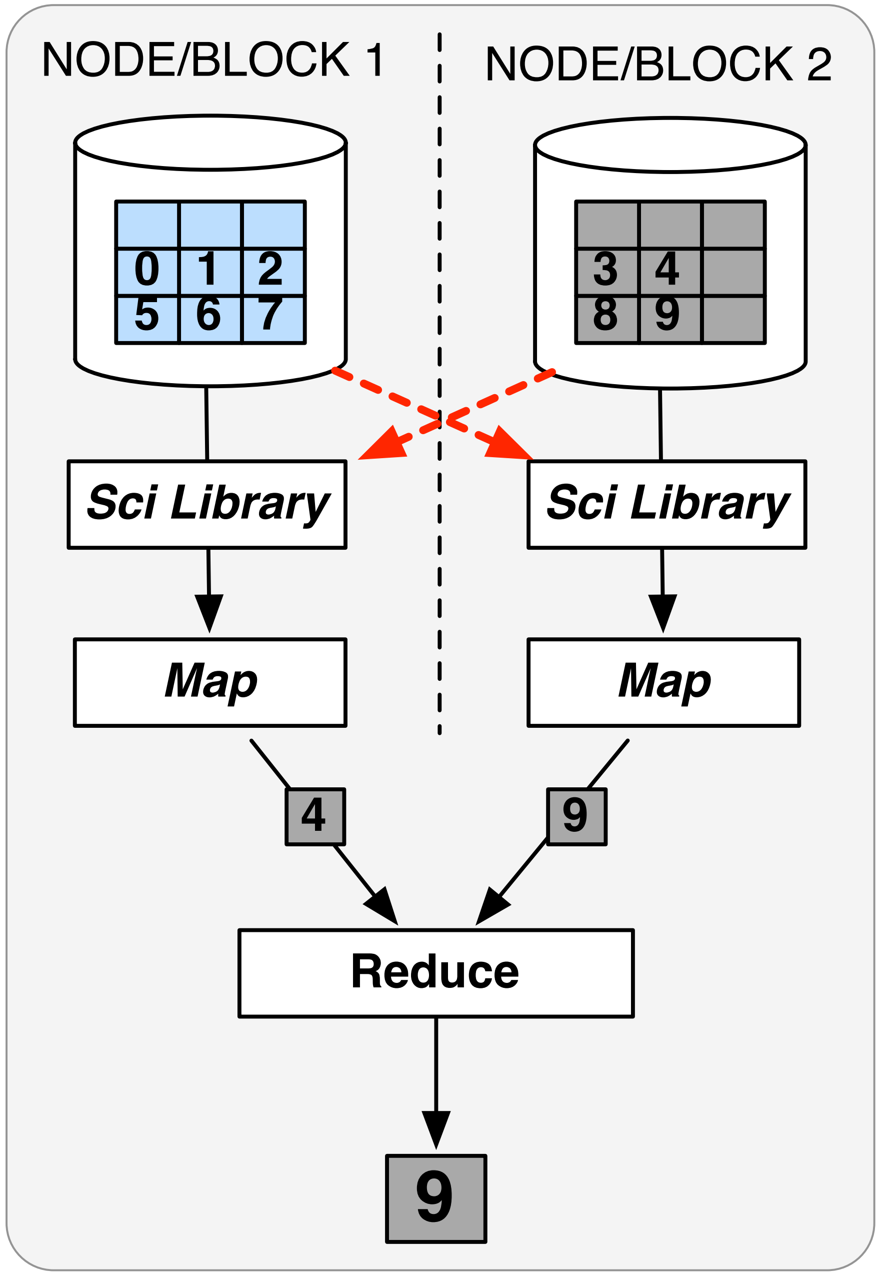


Naïve Partitioning

Logical Execution

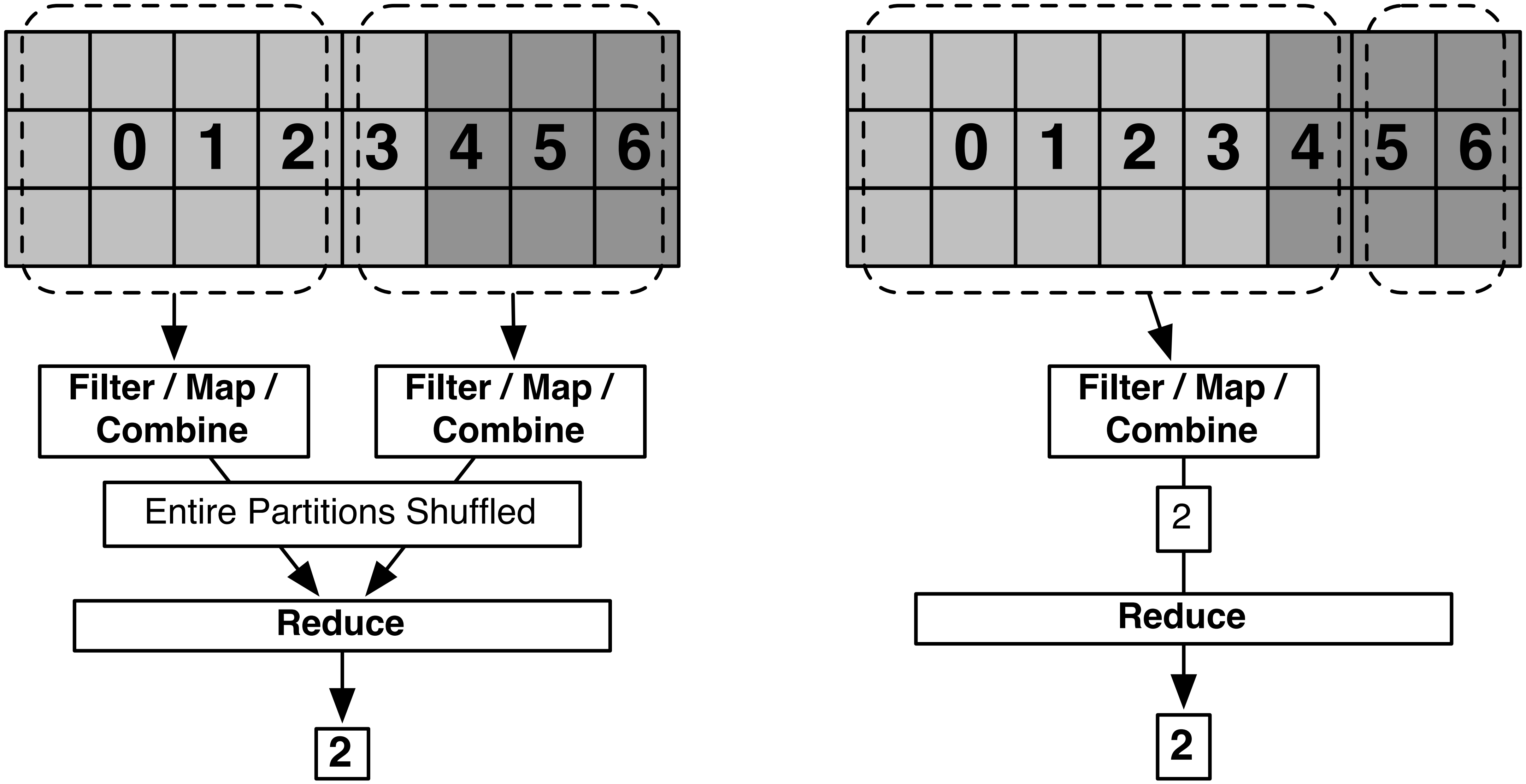


Physical Accesses



SciHadoop reduces data transfers

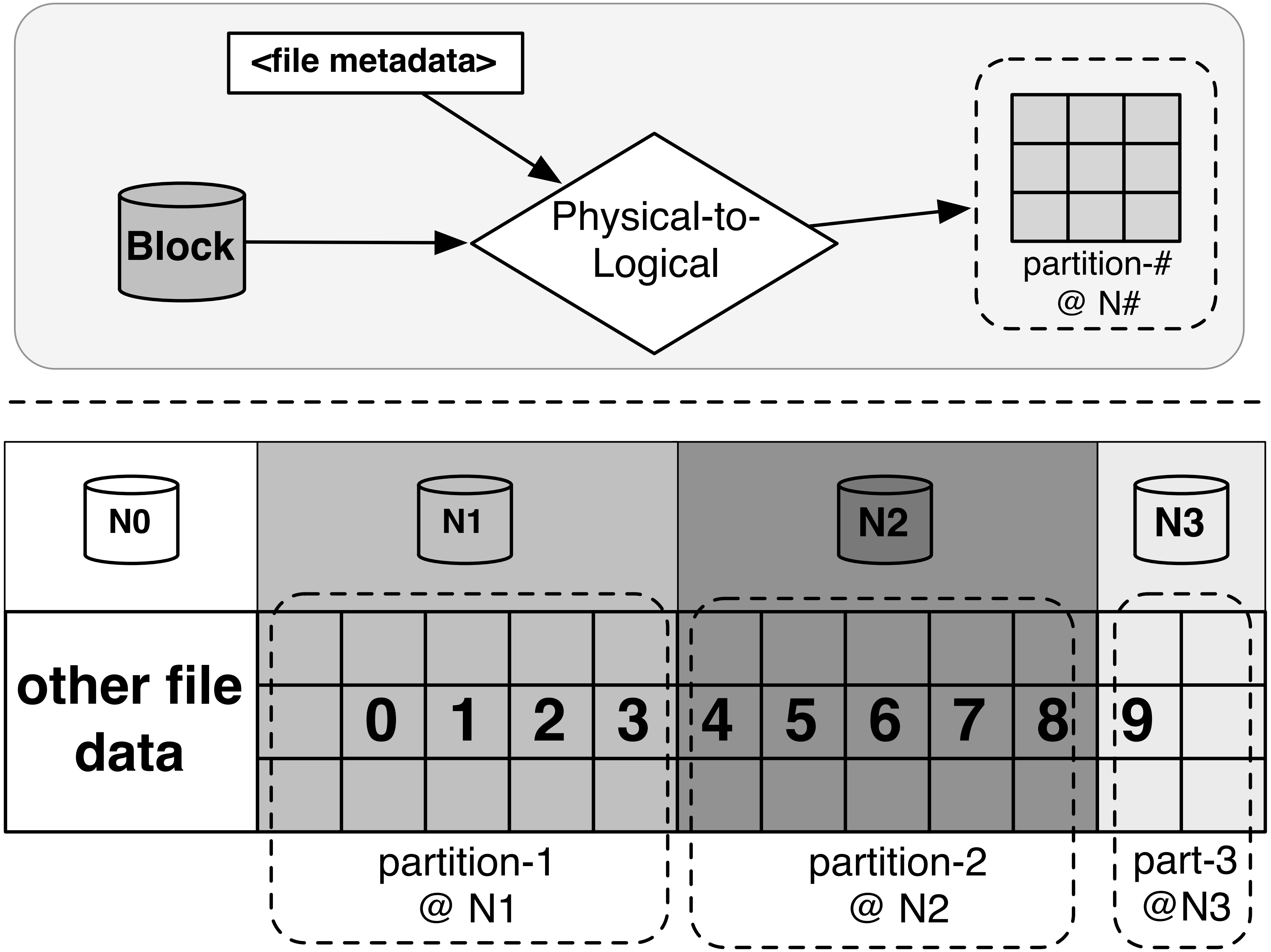
- Example: holistic functions (e.g. median)



(a) (b)

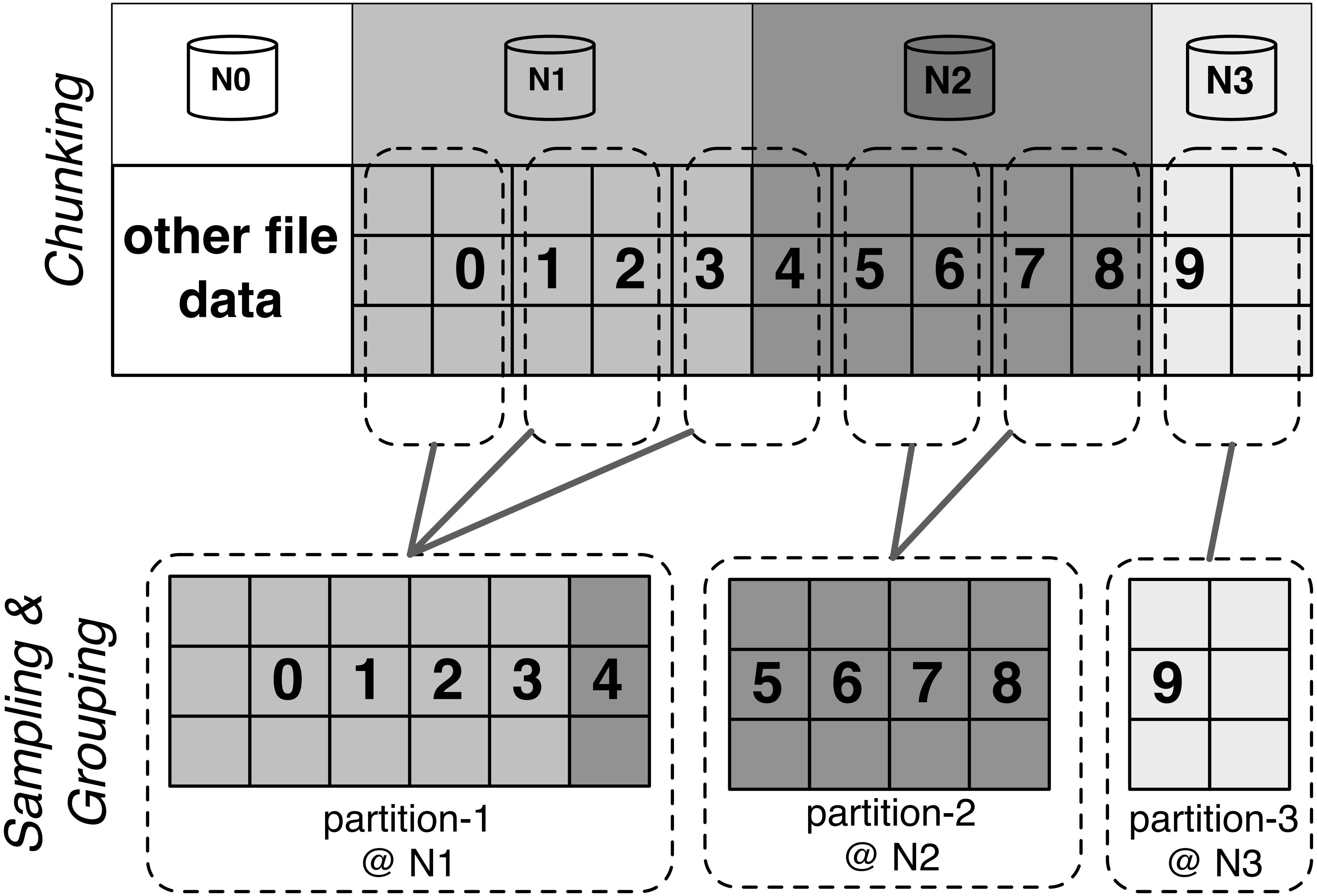
SciHadoop reduces remote reads

- Physical-to-logical translation (modified library)



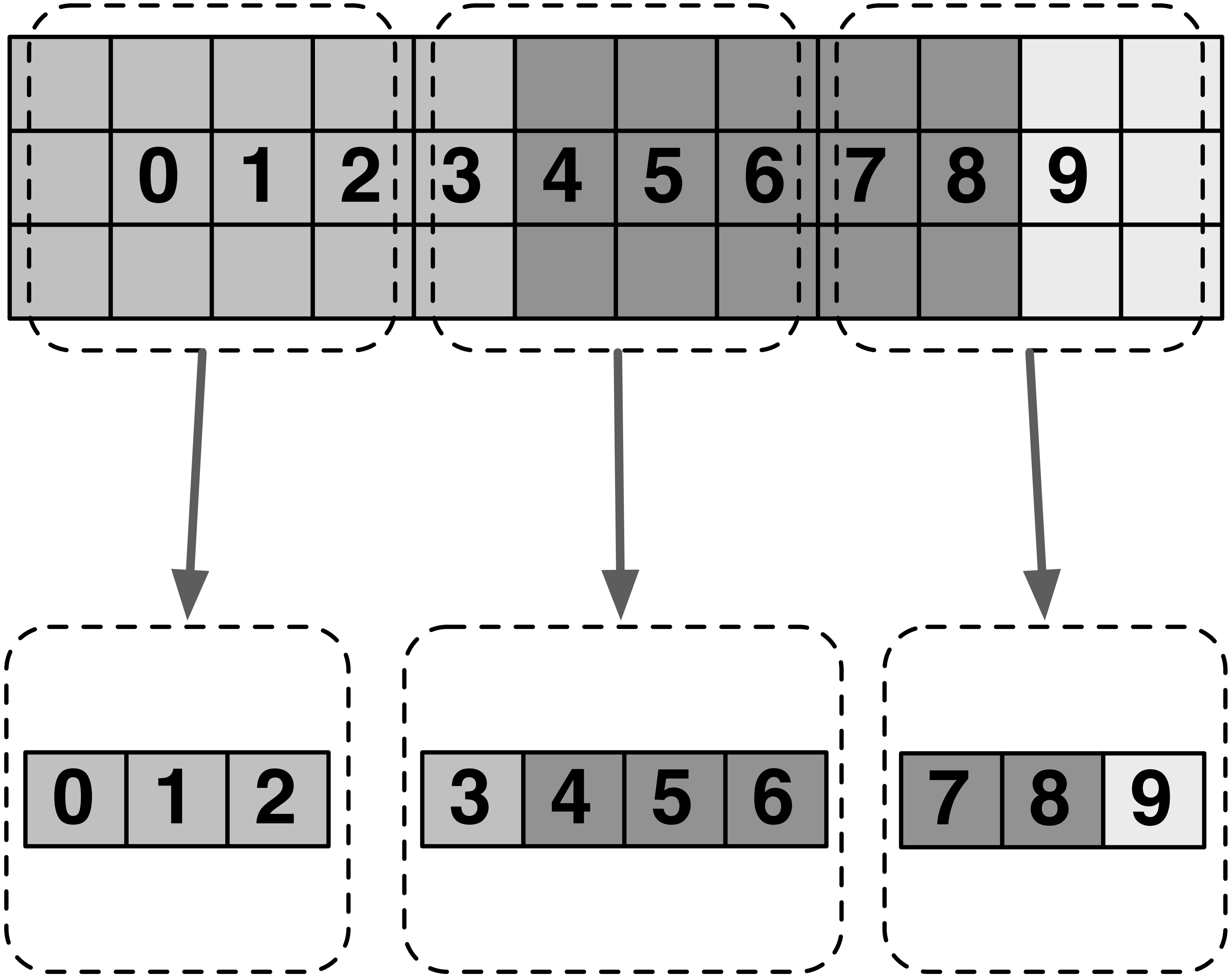
SciHadoop reduces remote reads

- Grouping chunks by sampling (unmodified lib)



SciHadoop reduces unnecessary reads

- Example: only access query’s data requirements



SciHadoop: Performance results

Test Name	Local Read (%)	Temp Data (GB)	CPU Util (%)	Run Time (Min)	Time σ (%)
First 4 use no Holistic Combiner					
baseline	9.3	2,586	34.7	159	14
baseline +NoScan	9.2	2,588	34.3	132	3
ChkGroup	80	2,608	24.3	188	10
PhysToLog	88	2,588	29.9	201	3
Next 4 use Holistic Combiner with baseline					
baseline	9.5	107	79.1	28	2
NoScan	9.5	107	80.7	27	3
+NoScan +*HaPart*	8.8	107	81.3	26	1
+*HaPart*	8.6	107	79.3	26	0.7
Next 3 use Holistic Combiner with Local-Read Optimizations					
ChkGroup +*HaPart* +NoScan	70.7	116	84.7	25	0.4
ChkGroup +NoScan	79.3	188	83.1	26	1
PhysToLog +NoScan	88.1	196	82.8	27	2

SciHadoop: Summary

- Map/Reduce data processing on scientific data using standard access libraries (here NetCDF3) and Hadoop
- Declarative query interface for Map/Reduce programs
- Powerful optimizations enabled by access to both logical and physical structure of data

More hierarchical? Presentation to user?

- Hierarchies and files are here to **stay**
  - Required for **grouping** data
  - Enhanced by **search** on attributes & relations
- Multiple views, multiple data models:
  - **Hierarchical** view for data groups and files
  - **Relational** view for catalog data
  - **Array**-based view for scientific data
  - **Graph**-based view for networking data
  - Full **integration** of all views and data models
  - **Declarative** access languages

Lessons from cloud storage?

- **Services** sell, capacity alone does not
  - E.g. safety, security, transcoding, archiving, compliance, elasticity, ...
- **Availability** and speed sell
  - Design around CAP Theorem
  - Understand consistency requirements
- What you can’t **meter**, you can’t sell
  - Price by SLOs, sell predictability
- **Failures** correlated (bursty) along failure domains
- **Key/value** stores: memcached, Cassandra, S3

Record-based IO?

- Now: Record-based IO above middleware
- Future: Record-based IO at OSD interfaces
  - Minimizes data movement
  - Supports also page-based interface for bulk IO

Crazy ideas

- Storage API: data objects in VTK Pipeline
- Scientific data curation as a game: why is Solitaire, essentially a sorting & assignment activity, so addictive?
- Jitter elimination by performance management. Then we can run everything everywhere.

Acknowledgements

UCSC: Joe Buck, Noah Watkins, Jeff LeFevre, Kleoni Ioannidou, Alkis Polyzotis, Sott Brandt, Wang-Chiew Tan

LANL: John Bent, Gary Grider, Meghan Wingate, James Nunez, Carolyn Connor, Lucho Ionkov, Mike Lang, Jim Ahrens

LLNL: Maya Gokhale, Celeste Matarazzo, Sasha Ames

UCAR/Unidata: Russ Rew

Thank you!

systems.soe.ucsc.edu